

FIG. 1 is a block diagram of a system architecture for resource allocation. The system includes a Player Agent (122) and a Resource Agent (124) connected via a bid (126) and an Allocation command (128). The Resource Agent is connected to an Accounting module (106) and a Network Ctl & Mgmt module (108). The Accounting module includes an IHN.db and a SOL. The Network Ctl & Mgmt module includes a SNMP and a COFS. The Resource Agent also includes a Strategy and a Valuation module. A GUT (110) is connected to the Resource Agent. A Protocol handler (102) is connected to the Player Agent. A legend defines the symbols: Agent (rectangle), Allocation Rule (circle), Strategy (triangle), Valuation (pentagon), GUT (smiley face), and Protocol handler (square).

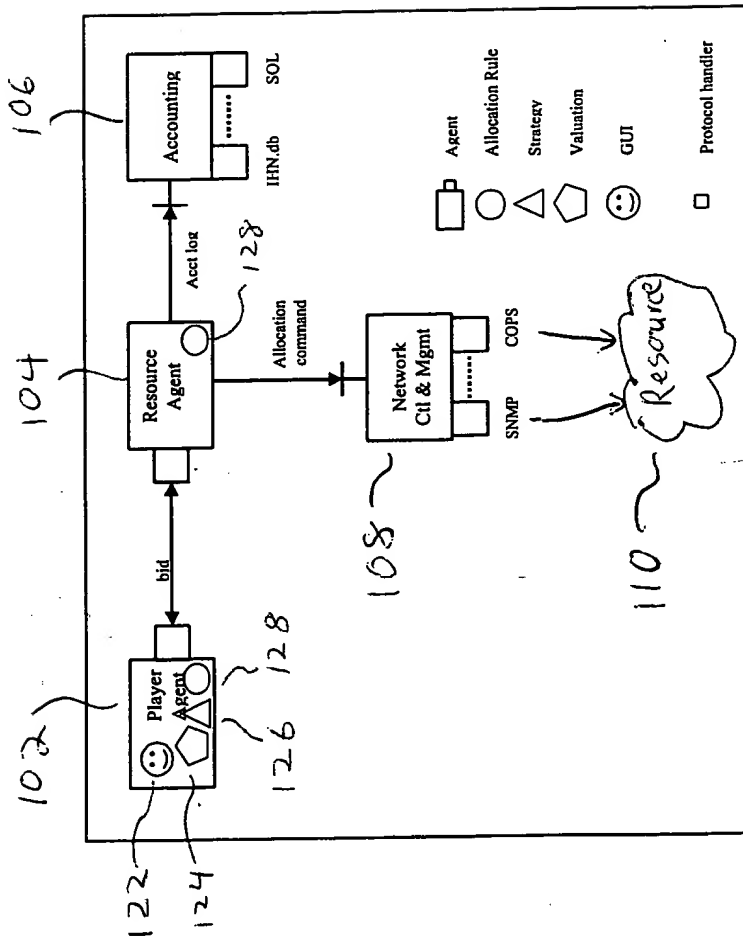


Fig 1

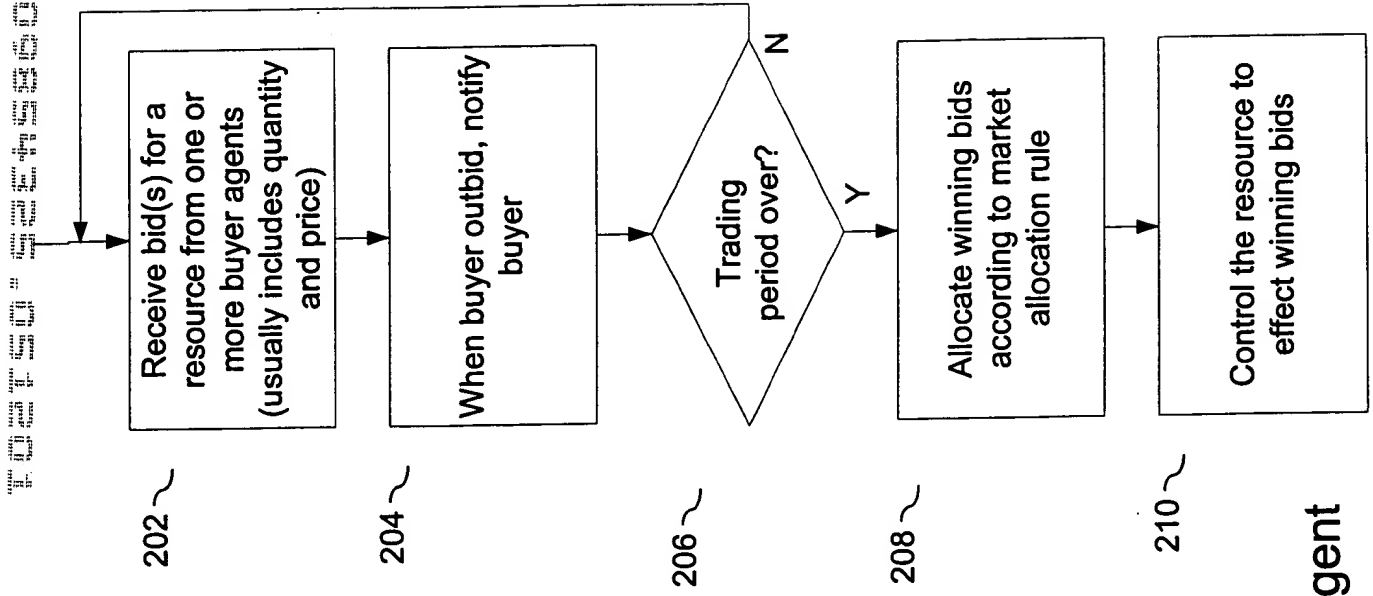


Fig. 2  
Resource Agent

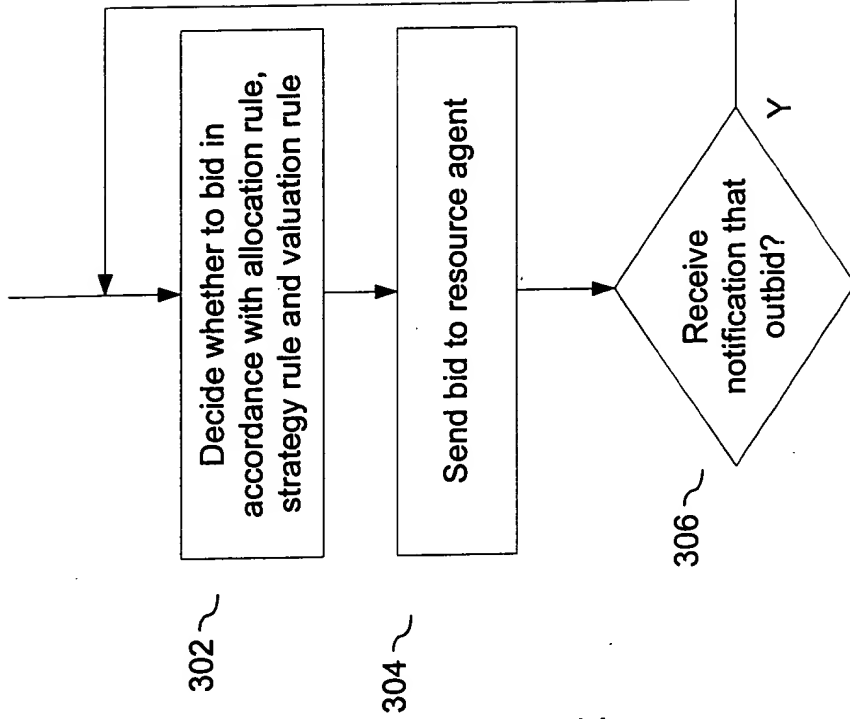


Fig. 3  
Player agent  
(Buyer)

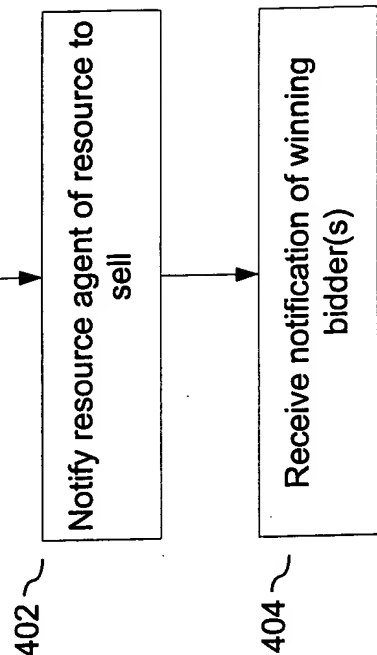


Fig. 4  
Player agent  
(Seller)

100 units of resource:  
A bids for 50 at \$3,  
B bids for 30 at \$2  
C bids for 30 at \$1  
D bid for 20 at \$0.50

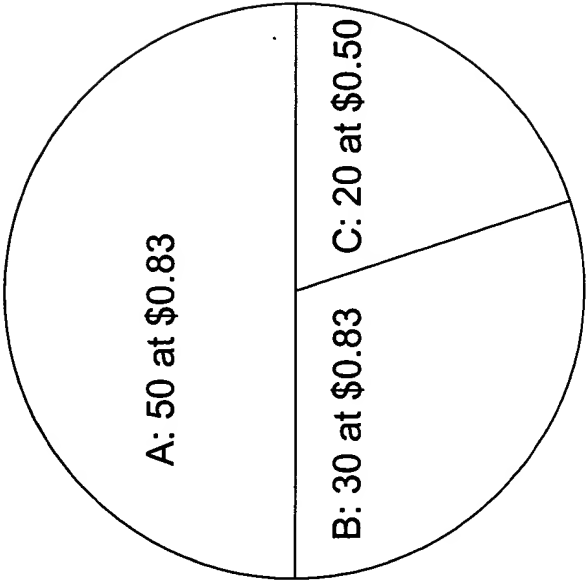


Fig. 5  
Example Market Allocation Rule  
(PSP)

Quantity	Price	Valuation
-	-	-
-	-	-
-	-	-
-	-	-

Fig. 6(a)  
Example  
Valuation Rule

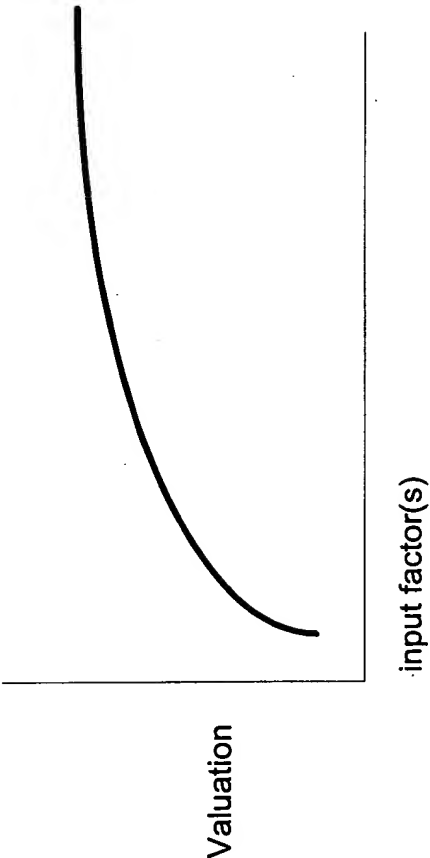


Fig. 6(b)  
Example  
Valuation Rule

If allocation rule is PSP  
[actions for PSP bidding]

If allocation rule is English auction  
[actions for English auction]

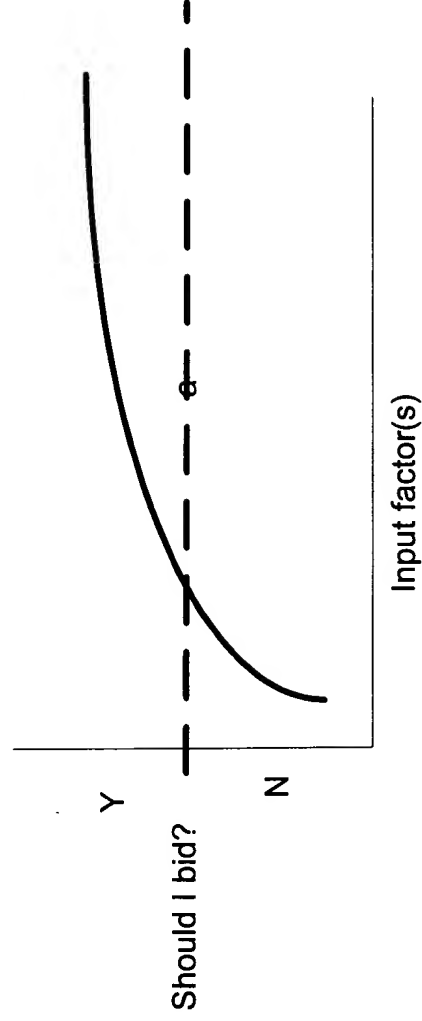


Fig. 7(a)  
Example Strategy  
Rule

Fig. 7(b)  
Example Strategy  
Rule

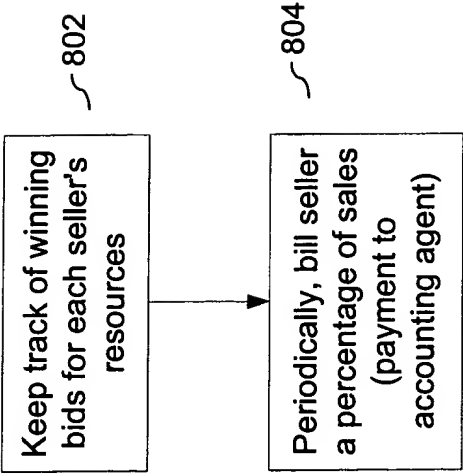


Fig. 8



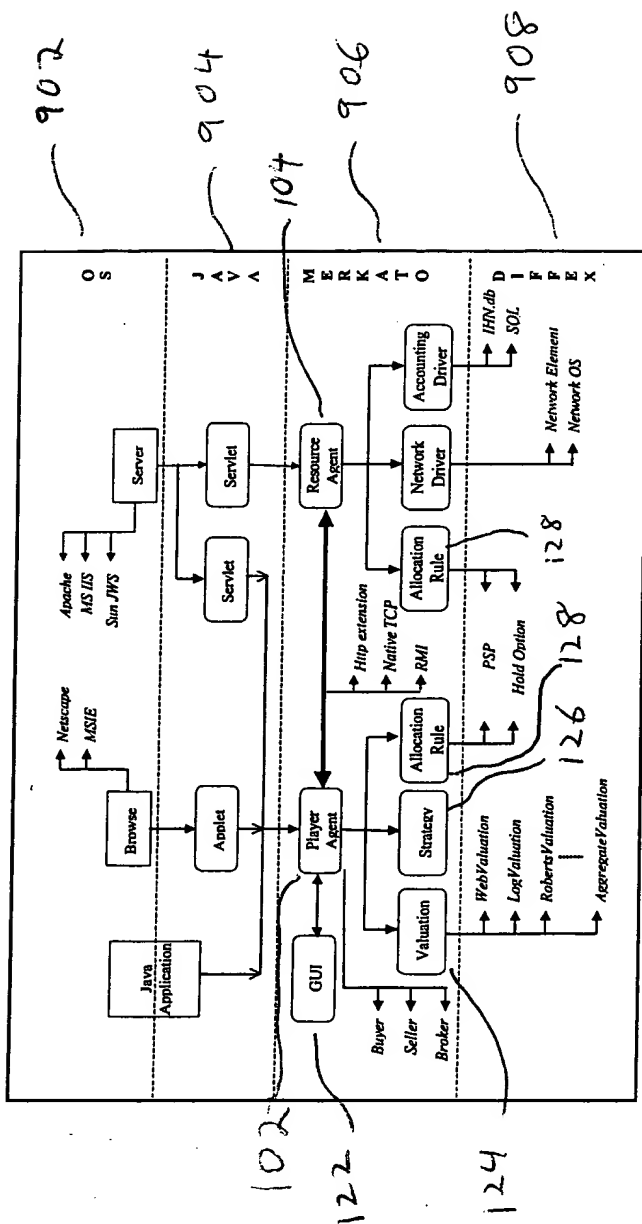


Fig. 9(a)

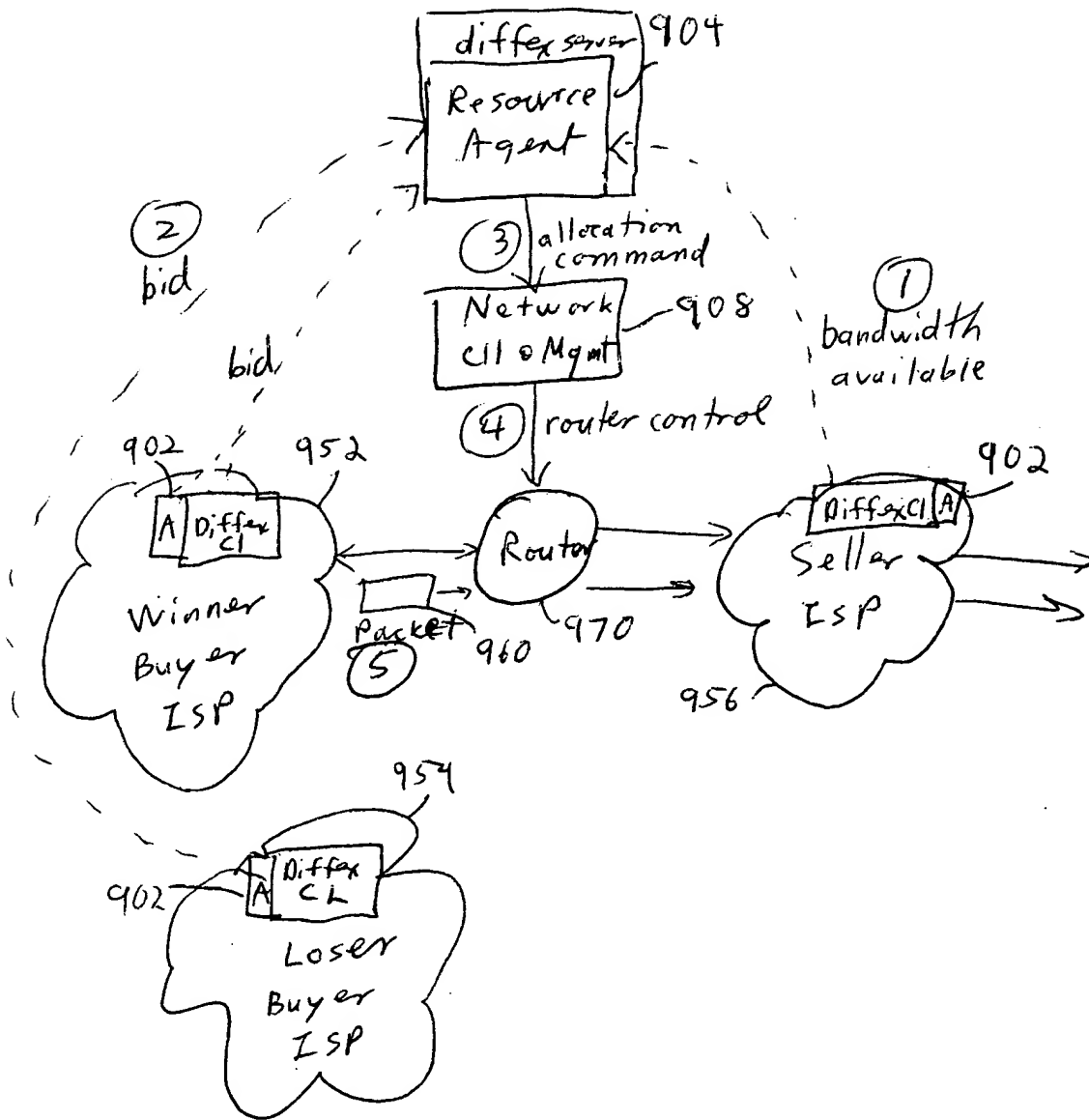
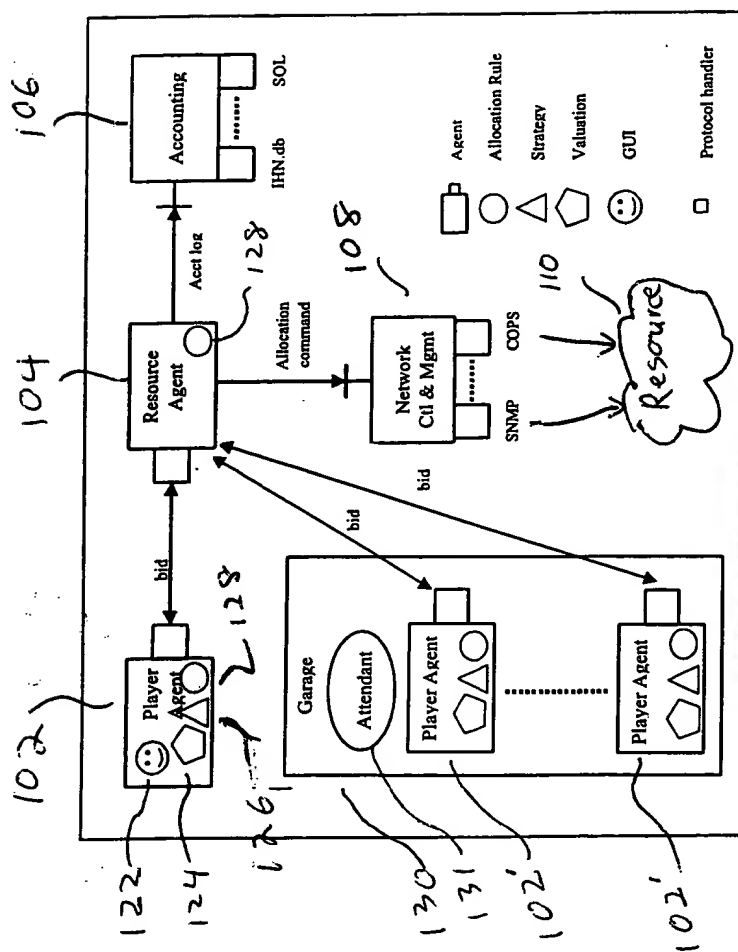


Fig 9(b)



10

```

<?xml version="1.0" encoding="UTF-8" ?>
- <AuctionPlayer context="http://HOSTNAME:HTTP_PORT/bx/garage">
- <SingleFrameGUI>
  <TextPanel name="News" height="50" visible="true" border="false" />
  <LoginPanel name="Login" height="160" visible="true" border="true" />
  <ResourceAgentPanel name="ResourceAgent" height="80" visible="true"
    border="true" />
  <UploadAgentPanel name="Garage" height="80" visible="true"
    border="true" />
  <BidCanvasPanel name="BidCanvas" height="180" visible="false"
    border="true" />
- <StrategyChoicePanel name="Strategies" height="160" visible="true"
  border="true">
  <StrategyPanel name="Manual" strategy="ManualStrategy" />
  <StrategyPanelNotEditable name="Auto" strategy="TruthfulStrategy" />
</StrategyChoicePanel>
- <ValuationChoicePanel name="Valuations" height="240" visible="false"
  border="true">
  <WebValuationPanel name="Web Valuation" valuation="WebValuation" />
  <ValuationPanel name="Elastic Demand" valuation="RobertsValuation" />
  <ValuationPanel name="Inelastic Demand" valuation="LinearValuation" />
  <BudgetValuationPanel name="Budget Valuation" label=""
    valuation="BudgetValuation" />
</ValuationChoicePanel>
  <PlayerInfoPanel name="Allocation" height="120" visible="true"
    border="true" />
  <BudgetPanel name="Budget" height="80" visible="false" border="true" />
  <DisplayPanel name="Units" height="80" visible="false" border="true" />
  <IPAddressPanel name="IP" height="110" visible="false" border="true" />
  <ConnectionPanel name="Connection" height="140" visible="false"
    border="true" />
  <BidTablePanel name="Bid Table" height="400" visible="false"
    border="true" />
  <BidGraphPanel name="Bid Graph" height="400" visible="false"
    border="true" />
  <AllocationGraphPanel name="Allocation Graph" height="400" visible="false"
    border="true" />
</SingleFrameGUI>
  <PlayerIdentity name="USERNAME" passwd="PASSWD" ipaddress="IP_ADDRESS"
    netmask="NETMASK" />
- <LinearValuation label="Inelastic Demand">
  <Parameter name="qmax" value="45000.0" label="Kbps" />
  <Parameter name="vmax" value="44928.0" label="$ /month" />
</LinearValuation>
- <RobertsValuation current="false" label="Elastic Demand">
  <Parameter name="qmax" value="45000.0" label="Kbps" />
  <Parameter name="vmax" value="4928.0" label="$ /month" />
</RobertsValuation>
- <BudgetValuation current="true" label="Budget Valuation">
  <Parameter name="qmax" value="1000.0" label="Kbps" />
  <Parameter name="budget" value="100.0" label="$ /month" />
</BudgetValuation>

```

11 (a)

- <WebValuation label="**Web Valuation**">  
 <param name="**delay**" value="**100.0**" />  
 <param name="**hitspermonth**" value="**100000.0**" />  
 <param name="**filesize**" value="**1000.0**" />  
 <param name="**centsperhit**" value="**0.1**" />  
 <param name="**randomize**" value="**false**" />  
</WebValuation>  
<Parameter name="**budget**" value="**51840.6**" label="**\$/month**" />  
<ManualStrategy current="**false**" label="**Manual**" />  
<TruthfulStrategy current="**true**" label="**Auto**" />  
<resourceAgentURL nickname="**RESOURCE\_NAME**"  
 current="**true**">**http://HOSTNAME:HTTP\_PORT/bx/RESOURCE\_NAME**</resourceAge  
<uploadURL nickname="**HOSTNAME**  
 **garage**">**http://HOSTNAME:HTTP\_PORT/bx/garage**</uploadURL>  
<param name="**playerInterval**" value="**2000**" />  
<param name="**timeout**" value="**2000**" />  
<param name="**timelabel**" value="**min**" />  
<param name="**currencylabel**" value="**c**" />  
<param name="**quantitylabel**" value="**Mbps**" />  
<param name="**debug**" value="**false**" />  
</AuctionPlayer>

```

/*
 * File:          Truthful.java
 *
 * Remark:        Strategy for player with diminishing returns
 *
 * $Id: Truthful.java,v 1.16          07:43:19 cobe Exp $
 *
 */

package ihm.merkato;
import org.w3c.dom.*;
import com.sun.xml.tree.XmlDocument;

/**
 * The strategy that bids the truthful best reply as in Proposition 1 o
 * f
 * the PSP paper.
 * It will only submit the bid if utility will be increased by at least
 * epsilon.
 * <p>
 * @author Nemo Semret
 * (
 */
public class Truthful extends AuctionStrategy {

    Bid tmp = createBid();
    /**
     * Finds truthful best reply as in Proposition 1 of
     * the PSP paper.
     * Sets the bid at the player if utility will be increased by at leas
     t
     * epsilon.
     * <p>
     * If timelogging is enabled, this will write to the player's
     * log a line with current time, bid, allocation, and utility,
     * at each call.
     * @see #epsilon
     * @see ihm.merkato.AuctionPlayer#setBid
     */

    public boolean bid() {

        double lq=0, uq= getPlayer().getValuation().qmax(), mq= (uq+lq)/2,
            dq = getPlayer().dq();

        if(debug()) {
            getPlayer().log("q range = ["+lq+
                "+uq+"] dq="+dq+
                " Q="+getPlayer().stuff());
            getPlayer().addnews(".");
        }
    }
}

```

12(a)  
Example of Agent strategy

```

// see Proposition 1
int i=0;
double mp, dv;
while( uq-lq > dq && i<20) {
    i++;
    mq = (lq+uq)/2;
    /*
        if(mq < getPlayer().stuff() -
            (getBidder().getBidList()).demandAtPrice(
                getPlayer().dval(mq, mq+dq),
                getPlayer().getId()))
        */
    // the following is equivalent and more general
    dv=getPlayer().getValuation().dval(mq, mq+dq);
    mp=getBidder().getBidList().marketPrice(getPlayer().stuff()
                                                -mq,
                                                getPlayer().getId());
};

if(debug())
    getPlayer().log("i="+i+" mq="+mq+" dv="+dv+" mp="+mp);

if(dv>mp)
    lq=mq;
else
    uq=mq;
}

tmp.bidderid = getPlayer().getId();
tmp.price = Data.MAXPRICE;
tmp.qty = lq;

if(debug())
    getPlayer().log("i="+i+" steps. q range = ["+lq
        +","+uq+"] currentbid="+
        getBidder().anteBid()+ " found "+tmp);

if(util(tmp) < 0) {
    uq= tmp.qty;
    lq=0;
}

i=0;
while(uq-lq>dq && i<20) {
    tmp.qty = (uq+lq)/2;
    i++;
    if(debug())
        getPlayer().log("i="+i+" q="+tmp.qty);
    if(util(tmp) < 0)
        uq= tmp.qty;
    else
        lq = tmp.qty;
}

```

12(b)  
Example of Agent Strategy

```

    }

    // need this in case the above loop is just outside the budget
    while (util( tmp) <0 && tmp.qty>0 && i <40) {
        i++;
        tmp.qty -=dq;
        if(debug())
            getPlayer().log("i="+i+" q="+tmp.qty);
    }

    if(debug())
        getPlayer().log(""+i+" steps. range=["+lq+", "+uq+"] currentbid="+
            getBidder().anteBid().qty);

1232 tmp.price = getPlayer().getValuation().dval(tmp.qty, tmp.qty+dq);

    double u = getPlayer().currentUtil();
    double newu = util(tmp) ;

    if(debug()) {
        getPlayer().log("currentalloc="+
            getBidder().currentAllocation()+
            " newbid="+tmp+" antebid="+
            getBidder().anteBid());
        getPlayer().log("u="+u+" newu="+newu+" ante="
            +util(getBidder().anteBid())
            +" fee="+getBidder().bidFee()
            +" epsilon="+epsilon()
            +" avgdur="+getAvgDuration());
    }

    if(getPlayer().trace()) {
        Bid alloc = getBidder().currentAllocation();

        getPlayer().log(""+getBidder().anteBid().qty
            +"\t"+getBidder().anteBid().price+"\t"+
            alloc.qty+"\t"+alloc.price+"\t"+
            getPlayer().currentUtil());
    }

1234 { if ( newu > u + epsilon()) {
    if(debug()) getPlayer().addnews("*");
    return getBidder().setBid(tmp.qty, tmp.price);
}
else {
    if(debug()) getPlayer().addnews("-");
    return false;
}
}
}

```

12(c)  
Example of Agent Strategy



```

/*
 * BidList object
 *
 * File:      PSPBidList.java
 *
 *
 *
 *
 *
 *
 */

```

```

package ihn.diffpex;

```

```

import ihn.merkato.Bid;
import ihn.merkato.Data;
/**

```

```

 *
 *
 */
public class PSPBidList extends ihn.merkato.GenericBidList {

    /**
     * Compute an allocation given the current profile of opponents
in
     * this bidlist. This class uses the Progressive-Second-Price
     * auction rule.
     * @param tb The bid for which the allocation is to be calculate
d.
     * @param Q The total quantity of resource available.
     */

```

```

    public Bid allocation(Bid tb, double Q) {
        return PSPAllocation(tb, Q);
    }

```

```

    /**
     * Compute an allocation given the current profile of opponents
in
     * this bidlist, with the Progressive-Second-Price
     * auction rule.
     * @param tb The bid for which the allocation is to be calculate

```

d.

```
* @param Q The total quantity of resource available.  
*/
```

```
private synchronized Bid PSpallocation(Bid tb, double Q) {
```

```
    Bid index = top;
```

```
    Bid alloc= new Bid();
```

```
    double leftover = Q; //leftover with player id
```

```
    double leftoverwo =Q; //leftover without player id
```

```
    double qAj,qAj0, num=0, den=0;
```

```
    boolean gotcha = false;
```

```
    alloc.qty = Math.min(tb.qty,
```

```
                        Math.max(Q-demandAtPrice(tb.price, tb.bid
```

```
derid),0));
```

```
    while(index.next != null) {
```

```
        index = index.next;
```

```
        if(index.bidderid != tb.bidderid) {
```

```
            if(index.price <= tb.price && !gotcha) {
```

```
                leftover -= tb.qty;
```

```
                if(leftover <=0)leftover=0;
```

```
                gotcha = true;
```

```
            }
```

```
            qAj = (index.qty <= leftover ? index.qty : leftover);
```

```
            qAj0= (index.qty <= leftoverwo ? index.qty : leftoverwo);
```

```
            num += (index.price* (qAj0- qAj));
```

```
            //      den += (qAj0- qAj);
```

```
            leftoverwo -= index.qty;
```

```
            leftover -= index.qty;
```

```
            if(leftover <=0)leftover=0;
```

```
            if(leftoverwo <=0)leftoverwo=0;
```

```
        }
```

```
    }
```

```
//    if (!gotcha) alloc.qty = (tb.qty <= leftover ? tb.qty :
```

```
leftover);
```

```
//    alloc.price = den>0 ? num/den : 0;
```

```
alloc.price = alloc.qty>0 ? num/alloc.qty : 0;  
alloc.bidderid = tb.bidderid;
```

```
return alloc;
```

```
}
```

```
/**
```

```
 * Bids with ID#0 are not counted.
```

```
*/
```

```
public double revenue(double Q) {
```

```
    Bid index = top;
```

```
    double r=0;
```

```
    int I=0;
```

```
    Bid al;
```

```
    while (index.next != null) {
```

```
        index = index.next;
```

```
        I++;
```

```
        if(index.bidderid!=0) {
```

```
            al = allocation(index,Q);
```

```
            r+= al.qty*al.price;
```

```
            //value(index.bidderid,Q);
```

```
        }
```

```
    }
```

```
    return r;
```

```
    //    if(I==0) return 0;
```

```
    //    else return r -= (I-1)*value(Data.NOBODY, Q);
```

```
}
```

```
}  
// substituted 8 float to double
```

11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000

```
<?xml version="1.0" encoding="UTF-8" ?>
- <GenericAuctionAgent
  context="http://HOSTNAME:HTTP_PORT/bx/RESOURCE_NAME">
    <PlayerIdentity name="RESOURCE_USER" passwd="RESOURCE_PASSWD"
      ipaddress="127.0.0.1" netmask="255.255.255.255" />
  - <PSPBidList>
    <param name="randomduration" value="false" />
    <param name="duration" value="60000" />
    <param name="mustconv" value="true" />
    <param name="bidfee" value="0.01" />
    <param name="capacity" value="20000.0" />
  </PSPBidList>
  <UnixCryptAuthenticator passwdfile="MERKATO_HOME/accounts/passwd" />
- <LinearValuation>
  <Parameter name="qmax" value="QMAX_VAL" label="QMAX_UNITS" />
  <Parameter name="vmax" value="VMAX_VAL" label="VMAX_UNITS" />
</LinearValuation>
<param name="accountingDriverClass"
  value="ihn.merkato.AccountManager" />
<param name="accountFile"
  value="http://HOSTNAME:HTTP_PORT/bx/dbstub" />
<param name="hwDriverClass" value="RESOURCE_DRIVER_CLASS" />
<param name="hwDevice" value="RESOURCE_DRIVER_INIT" />
<param name="maxNBids" value="100" />
<param name="verbose" value="true" />
<param name="rememberIds" value="false" />
<param name="clientTimeout" value="60000" />
<param name="serverTimeout" value="30000" />
<param name="pause" value="5000" />
<param name="detailedlog" value="true" />
<Parameter name="maxBidFee" value="1.0" label="$" />
<Parameter name="maxAccountBalance" value="10000.0" label="$" />
</GenericAuctionAgent>
```

Fig 14

## HTML – Not Bidding

Note: All changes require that "Submit" be pressed to send change to agent in "garage"

Select "active" to begin bidding

"Budget" is used to calculate price per unit bandwidth bid during auction. (Must enable "active" first). User is encouraged to bid high during periods of heavy use and bid low, or not at all during periods of light use.

"Refresh" updates screen display

"Submit" sends new values to agent in "garage" and exits

Select the units for the display. Previously entered values will scale to the new units

There is no cost accrued to buyers who are not bidding. They will be placed in the best-effort queue until they elect to bid for bandwidth.

"Submit" updates the budget value of the garaged agent to what you have entered into this screen and exits. At this point, it exits to a generic Merkato screen, but for customers, it will exit to a StreamingHand page.

Fig 15(a)

# HTML - Bidding

Note: All changes require that "Submit" be pressed to send change to agent in "garage"

The screenshot shows the 'Merkato Auction Buyer' window. It contains a table with the following data:

Field	Value
Budget	1000 \$/day
EA	400.0 Kbps
Allocation	400.0 Kbps
Round Duration	60.00 sec
Primary Unit	Day
Secondary Unit	Day
Quantity Unit	Kbps

Below the table are two buttons: 'Refresh' and 'Submit'. Annotations point to various elements:

- Select "Inactive" to stop bidding**: Points to the top right corner of the window.
- Budget (same as in "inactive" screen)**: Points to the 'Budget' field.
- This is the last price offered with quantity desired bid (changes often so need to "Refresh")**: Points to the 'EA' field.
- Amount of bandwidth and extended price allocated to this agent during the last auction round**: Points to the 'Allocation' field.
- Units (same in in "inactive" screen)**: Points to the 'Primary Unit' and 'Secondary Unit' fields.
- Duration of last bid round. This time varies based on how active the bidding is during a round.**: Points to the 'Round Duration' field.
- Refresh (same in in "inactive" screen)**: Points to the 'Refresh' button.
- Submit (same in in "inactive" screen)**: Points to the 'Submit' button.

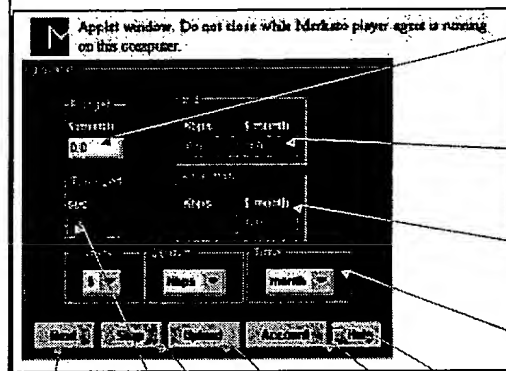
User will generally want to bid high during period of heavy use and lower during periods of light use.

The agent will attempt to obtain as much bandwidth as possible without exceeding budget. Conversely, the agent will request smaller and smaller amounts of bandwidth until they can obtain something for the budget price.

"Refresh" updates the screen. It does not send any changes to the "budget" value to the garaged agent. "Submit" does this (and then exits).

Fig 15(b)

## Wizard – Bid Window



"Budget" is used to calculate price per unit bandwidth bid during auction.

This is the last price offered with quantity desired bid (changes often so need to "Refresh").

Amount of bandwidth and extended price allocated to this agent during the last auction round

Select the units for the display. Previously entered values will scale to the new units

Display help. When checked, mouse-button presses bring up help rather than performing function

Show auction graph

Stop bidding

Countdown timer for current auction. Reset whenever a new bid is received.

Uploads agent to garage where it can continue bidding and exits

Display account summary screen

"Stop" means to stop bidding. This bidding agent will not be charged and they will be placed in the shared best-effort queue.

"Upload" uploads the configuration to the garaged agent. Not that this will change some advanced settings to those assumed by this simple valuation and strategy model.

This simple budget-based valuation model has the bidding agent attempt to get as much bandwidth as possible without exceeding the budget number.

The strategy is based on the formula:  $\text{price-per-unit-bandwidth} * \text{bandwidth-allocated} = \text{total-price-paid}$

Where the total-price-paid ("budget") is held constant, and the other two variables allowed to be altered.

Following this strategy, the bidder will first attempt to get all the bandwidth the seller is offering for their budgeted amount, which works out to the lowest possible price-per-unit-bandwidth. If unsuccessful, the bidding agent gradually increases the offered price-per-unit-bandwidth and decreases the desired amount of bandwidth, until they successfully win an allocation.

If all bidders follow this valuation model, they will each get a bandwidth allocation that is the same proportion to total bandwidth as their budget is to the combined budgets of all bidders.

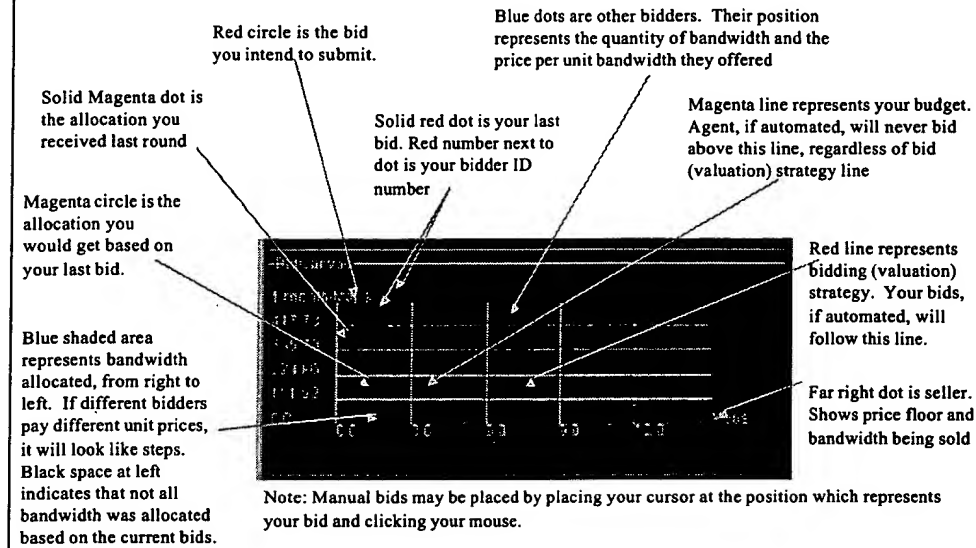
From the "Help" screen"

- Press **Start** to tell your agent to start bidding for you.
- Press **Stop** to tell your agent to stop.

Fig 15(c)

## “Bid Canvas” Screen

A dynamic display of the second price auction in progress



Red Valuation Line and Magenta Budget line are superimposed when “Budget” valuation is being used (default for “HTML” and “Wizard” Agent Interfaces).

Often the Red circle, red dot and magenta circle will be very close together.

Fig 15(d)



## Status bar

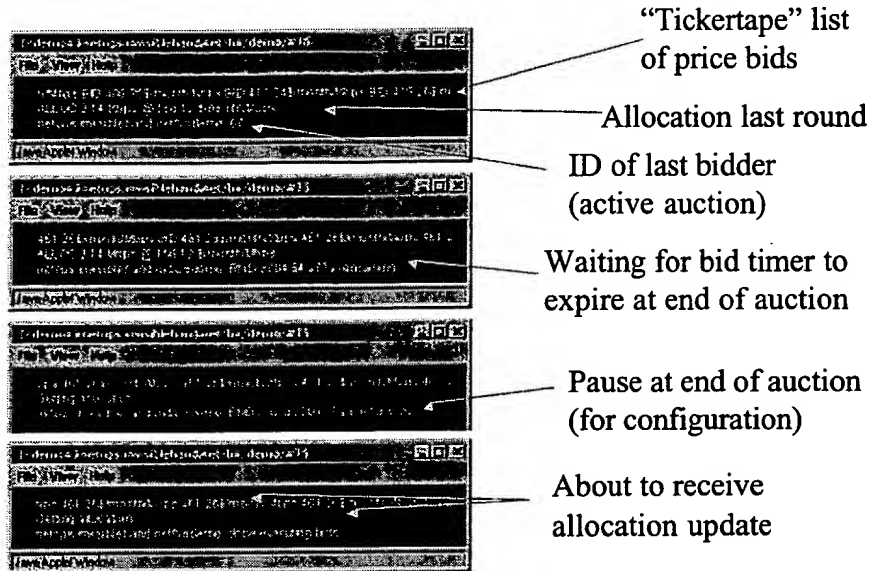
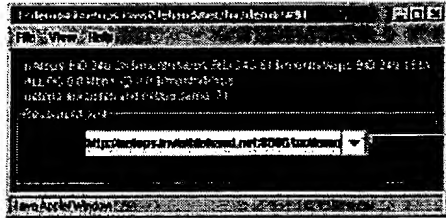


Fig 15(e)

## "Resource Agent" Subscreen

Selection screen for resource for which you are bidding



Pull-down menu allows you to determine which resource you would like to bid on.

Fig 15(f)

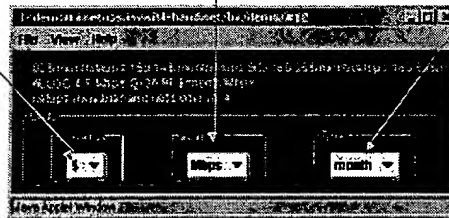
1. The user selects a resource from the pull-down menu.

## "Units" Subscreen

Enter the units for currency which you would like in all displays (currently, the only option is "\$")

Enter the units for bandwidth which you would like in all displays (options are "Kbps", "Mbps", "Gbps")

Enter the units for time which you would like in all displays (options are "ms"- millisc, "sec", "min", "hr" – hours, "day", "or "month").



Note: If you change units, any numerical values in any other subscreen will automatically be scaled to reflect the units change, but represent the same quantity as originally specified.

Fig 15(9)

## "Budget" subscreen

Selection screen for bidding "strategy"

You enter the "maximum cost run rate" here - this supercedes any higher values that might be derived from valuation curves. In other words, bids - which consist of a price per unit bandwidth and a total bandwidth desired - will not be placed if they would result in a greater price than that indicated here.

When the valuation type is "Budget", the "price per unit time" field cannot be altered because it is a duplicate of that entered in the valuation subscreen.

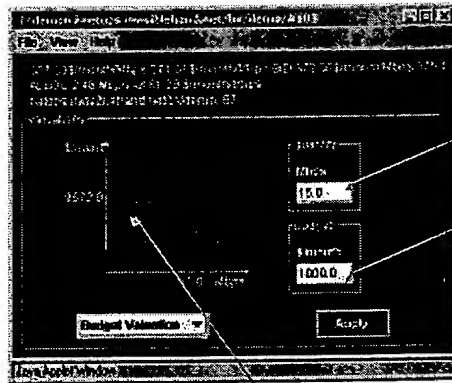
The screenshot shows a window titled "Budget" with a menu bar (File, View, Help) and a toolbar. The main area contains a "Maximum cost run rate" field with a value of 1000.0 and an "Apply" button. A line from the text above points to this field.

The screenshot shows a window titled "Budget" with a menu bar (File, View, Help) and a toolbar. The main area contains a "Price per unit time" field with a value of 100.0 and an "Apply" button. A line from the text above points to this field.

Fig 15(h)

# Valuations – Budget Valuation

Selection screen for bidding “strategy”



Maximum quantity of bandwidth desired (by default it is all the seller is offering)

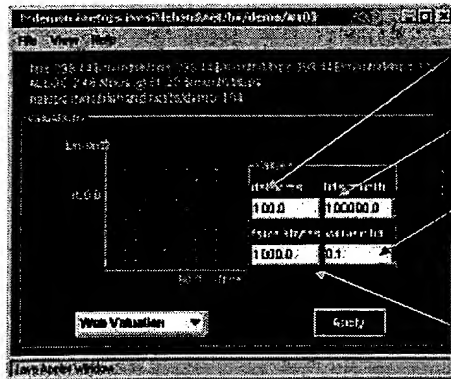
Maximum amount you are willing to pay for that bandwidth

This curve represents the value you place on bandwidth based on the amount you receive. The “budget” valuation curve represents a desire to get the maximum amount of bandwidth for a constant price. The “Valuation” curve in the bid canvas as your bid strategy is derived from the change in slope of this curve.

Fig 15(i)

# Valuations – Web Valuation

Selection screen for bidding “strategy”



Average desired delay in ms to transfer a file of the size indicated.

Number of such files expected to be downloaded per month

Value to you in cents per file downloaded

(Note: You can independently set your maximum monthly budget via the “Budget” screen, so the shape of this curve is more important than its maximum price-point)

Average size of file downloaded

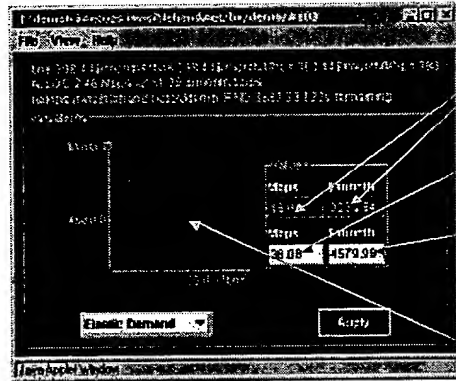
The web valuation attempts to translate a content hosting business need into bandwidth and price requirements. The formulas used are:  $(\text{max bandwidth in Mbps}) = \text{fsize} * 8 / \text{delay}$

$(\text{Max price in \$/month}) = \text{value} * (\text{hits per /month}) / 100$

Fig 15(j)

# Valuation – Elastic Demand

Selection screen for bidding “strategy”



This display provides the user a feel for the shape of the curve. The right display is the price-point for the amount of bandwidth to the left.

Max bandwidth desired (see discussion, below, for impact of this setting)

Max price-point (see discussion, below, for impact of this setting)

Note : You can independently set your maximum monthly budget via the “Budget” screen, so the shape of this curve is more important than its maximum price-point

Note 2: You may enter new values by dragging and dropping the red dot on the graph with your cursor

“Elastic” valuation models how users have historically valued internet bandwidth. The formula, when used as a bid strategy (see Bid Canvas), is:

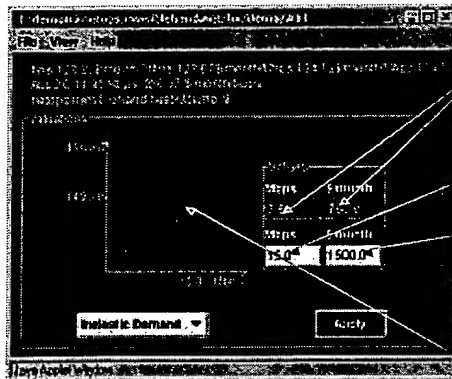
$$(\text{Price per unit bandwidth}) * (\text{Qty of Bandwidth})^2 = (.0012) * (\text{Max price-point}) * (\text{Max bandwidth})^2$$

Note 3: Constant (.0012) is correct for units shown, above. It will scale depending on units selected.

Fig 15(k)

# Valuation – Inelastic Demand

Selection screen for bidding “strategy”



This display provides the user a feel for the shape of the curve. The right display is the price-point for the amount of bandwidth to the left.

Max bandwidth desired (see discussion, below, for impact of this setting)

Max price-point (see discussion, below, for impact of this setting)

Note : You can independently set your maximum monthly budget via the “Budget” screen, so the shape of this curve is more important than its maximum price-point

Note 2: You may enter new values by dragging and dropping the red dot on the graph with your cursor

“Inelastic” valuation indicates that you wish to pay the same price per unit bandwidth regardless of the amount of bandwidth received. This results in a horizontal bid strategy line on the Bid canvas, following the formula:

$$(\text{Price per unit bandwidth}) = (\text{Max price-point}) / (\text{Maximum Bandwidth Desired})$$

When the elastic bid strategy is combined with the knowledge of the second price auction mechanism, it results in the following behavior:

If the elastic valuation is above the budget line, the agent will do a reverse calculation to determine when it can bid on the valuation line, but obtain the bandwidth on the budget line.

If the elastic valuation is below the budget line, the agent will continue to ask for the maximum amount of bandwidth at the valuation price and not accept a lesser amount of bandwidth.

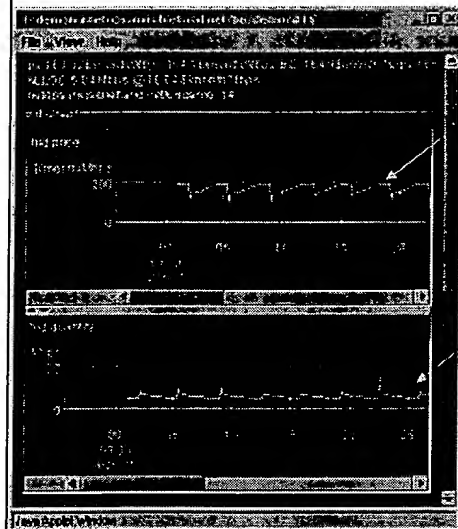
Fig 15(1)





## "Bid Graph" Subscreen

Bid history over time



Bid price (per unit bandwidth) over time. Cycles correspond to bidding rounds. Graph starts when subscreen is activated.

Quantity requested per unit time

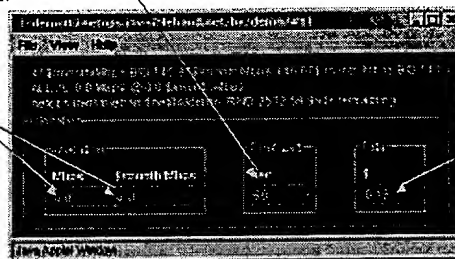
Fig 15(n)

## “Allocation” subscreen

Results of previous bidding round

If, as is normal, the bidding round is terminated when no bids are received within a configured amount of time, the “Time Left” counter will count down from the configured time, but get reset whenever a bid is received by the Resource Agent, from anyone. When this counter reaches zero, an allocation will be made and a new bidding round will begin after a slight pause to implement the allocation.

Allocation  
(Quantity and  
Price per unit  
bandwidth)  
received during  
the last bid cycle

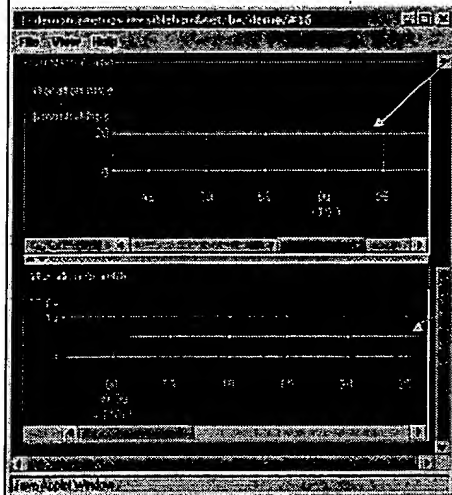


Total amount  
spent during  
this session

Fig 15(0)

## “Allocation Graph” Subscreen

Allocation history over time



Allocation price (per unit bandwidth) over time. Updated at the end of each bidding round. Graph starts when subscreen is activated.

Quantity received per unit time

Fig 15(p)

# 

A dynamic display of the second price auction in progress

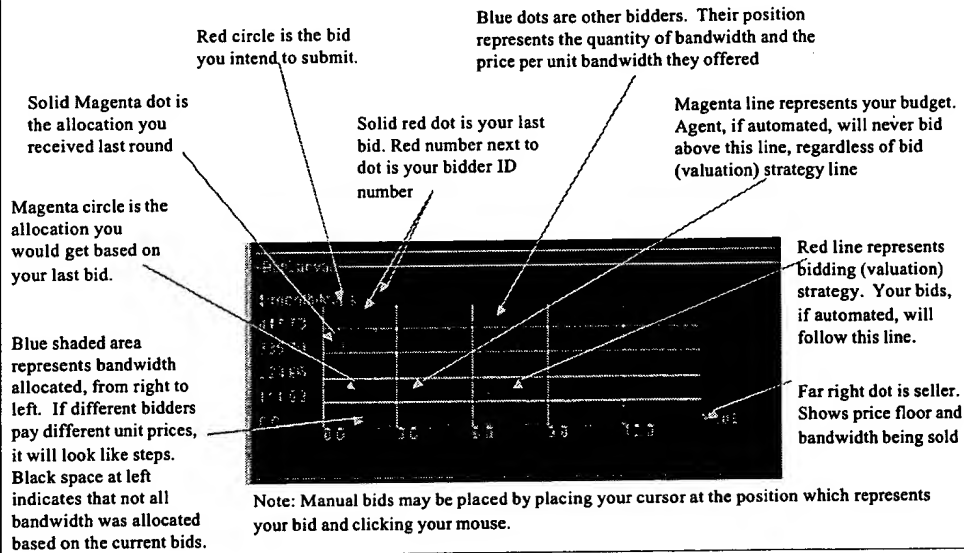


Fig 15(9)

## "Bid Table" Subscreen

A dynamic display of the second price auction in progress

ID's of  
bidders

Columns can be resized by  
dragging column separators

"Rate" is allocated Quantity times  
bid Price (per unit bandwidth)

Bidders shaded in blue would  
receive an allocation of bandwidth  
if no further bids were received

You are bidder with red text

Bidders with no shading would receive no  
allocation if all bids remained the same.

Bidders shaded in yellow are those used to  
calculate the auction price of bandwidth  
received by the bidder shown in red (you)

This is the "rate" bidder in red (you)  
would pay for the bandwidth allocated  
(as opposed to what you bid, above)

Bottom un-shaded bidder is the seller.  
The seller's "bid" is his price floor

ID	Quantity	Price	Rate
1	100	10.00	1000.00
2	100	9.00	900.00
3	100	8.00	800.00
4	100	7.00	700.00
5	100	6.00	600.00
6	100	5.00	500.00
7	100	4.00	400.00
8	100	3.00	300.00
9	100	2.00	200.00
10	100	1.00	100.00
11	100	0.00	0.00

Fig 15(n)